International Journal of Innovative
Computing, Information and Control
Volume **10**, Number **4**, August **2014**

 ISSN 1349-4198
pp. 1369–1379

# COLOR IMAGE RETRIEVAL UTILIZING EXTENDED FAST VQ CODEWORD SEARCH TECHNIQUE AND VECTOR COMPOSITION-BASED FEEDBACK

Zhi-Wei Zhang[1], Ming-Hui Wang[2], Zhe-Ming Lu[2,*] and Ting-Yuan Nie[1]

[1]School of Science
Qingdao Technological University
No. 11, Fushun Road, Qingdao 266033, P. R. China
shzhw@qtech.edu.cn

[2]School of Aeronautics and Astronautics
Zhejiang University
No. 38, Zheda Road, Hangzhou 310027, P. R. China
*Corresponding author: zheminglu@zju.edu.cn

Received September 2013; revised January 2014

Abstract. *This paper presents a novel fast image retrieval algorithm based on extended fast VQ codeword search (EFVQCS) technique to improve the efficiency of the content-based image retrieval (CBIR) system. The proposed scheme can effectively reduce the retrieval time, without decreasing the accuracy of retrieving the first K most similar images. To make each feature component in a feature vector within the same range, the Gaussian normalization technique is adopted. Furthermore, a simple vector composition-based relevance feedback method is utilized to improve the retrieval precision. Experimental results demonstrate that our scheme can not only speed up the retrieval process but also achieve satisfactory performance in precision, especially for large image databases with a high feature vector dimension.*

**Keywords:** Content based image retrieval, Vector quantization, Extended fast VQ codeword search, Gaussian normalization, Relevance feedback

1. **Introduction.** With the development of multimedia and Internet technologies, the amount of visual information over the Internet has grown exponentially. This brings the research heat in content-based visual information retrieval (CBVIR). As a significant branch of CBVIR, content-based image retrieval (CBIR) has gained the attention of many researchers all over the world. Recently, a large number of prototypes and practical products have been already developed. In a typical CBIR system [1], images are not manually annotated by keywords but described by their own visual features, including color features [2], texture features [3,4] and shape features [5], which are more natural and more suitable to the human visual system than the keywords used in traditional text-based image retrieval systems.

Currently, many existing CBIR schemes are performed on uncompressed domains. However, because of the huge amount of visual information, retrieval of compressed images is one of our needs in the information era. Since the compressed code of an image can be viewed as its compact description, recently, many researchers have made greater efforts in image retrieval based on compressed-domains such as vector quantization (VQ) [6,7], DCT [8], DWT [9], fractal coding [10], and block truncation coding (BTC) [11]. Although VQ [12] is an effective lossy image compression technique, it can be also used as a feature extractor to describe an image based on a codebook or used as a clustering method to quantize the feature space consisting of many feature vectors. The main idea

of VQ is to divide an image into blocks (vectors) and then encode them vector by vector using the indices of their nearest codewords in the predesigned codebook. In VQ-based retrieval schemes, features can be extracted from the index table of the spatial-domain VQ compressed image [6] or from the individual VQ codebook that is generated from the image [7]. The histogram of the VQ-compressed index sequence proposed in [13] is used to describe the features, while DCT-domain VQ index histograms are extracted as features from the image in the YCbCr color space in [14]. In addition, two new compressed-domain features named Multi-Stage Vector Quantization Index Histograms and Mean-Removed Vector Quantization Index Histograms for color image retrieval based on the YCbCr color space are proposed in [15].

In this paper, we consider another application of VQ in image retrieval, i.e., extending a fast codeword search algorithm [16] used in VQ image coding to speed up the similarity comparison process. The key contribution lies in enlarging the application field of VQ while solving the key search problem in fast image retrieval. Our proposed scheme can effectively reduce the retrieval time, without decreasing the accuracy of the retrieval results. In addition, to improve the performance in precision and recall, a simple relevance feedback scheme based on recomposing the query feature vector out of the retrieved results is proposed. The remainder of this paper is organized as follows. In Section 2, we introduce some related techniques, including the vector quantization technique, the fast codeword search principle and the Guassian normalization technique. Section 3 detailedly describes our fast image retrieval scheme. Section 4 performs some experiments to demonstrate the effectiveness of the proposed scheme. Section 5 concludes the whole paper.

## 2. **Preliminaries.**

In this section, we first give a brief introduction to vector quantization, since it is the origin of the fast codeword search algorithm. Then, the fast codeword search principle is introduced, which can be generalized to our fast retrieval scheme. Finally, the feature vector construction method based on Gaussian normalization is introduced.

### 2.1. **Vector quantization.**

Vector quantization (VQ) [12] is a famous block-based image encoding technique. As shown in Figure 1, VQ maps the $n$-dimensional Euclidean space $R^n$ into a finite subset $C = \{\boldsymbol{c}_i | i = 0, 1, \ldots, M-1\}$, which is generally called a codebook, where $\boldsymbol{c}_i$ is a codeword and $M$ is the codebook size. Before compression, VQ generates a representative codebook from a training set consisting of a number of training vectors. The image to be encoded is first divided into non-overlapping $n$-dimensional vectors. As shown in Figure 2, in the encoding stage, each $n$-dimensional input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is compared with each codeword in the codebook $C$ to find the best matching codeword

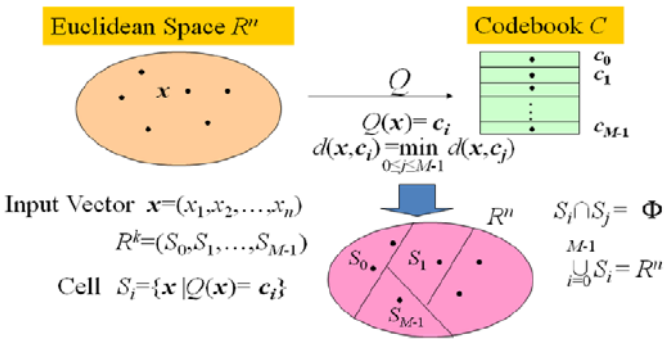


FIGURE 1. The principle of VQ

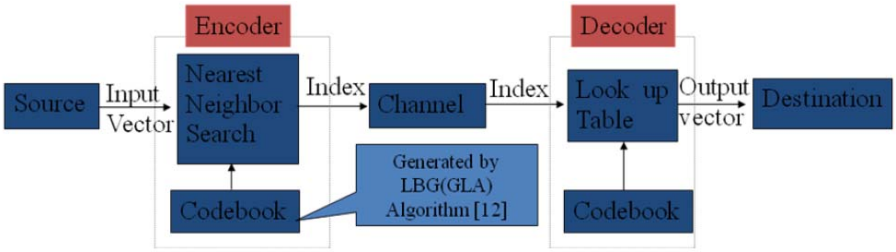


FIGURE 2. The encoding and decoding processes of VQ

$\boldsymbol{c}_i = (c_{i1}, c_{i2}, \ldots, c_{in})$, i.e.,

$$d(\boldsymbol{x}, \boldsymbol{c}_j) = \min_{0 \le j \le M-1} d(\boldsymbol{x}, \boldsymbol{c}_j) \tag{1}$$

where $d(\boldsymbol{x}, \boldsymbol{c}_j)$ is the squared Euclidean distance from the input vector $\boldsymbol{x}$ to the codeword $\boldsymbol{c}_j$. After encoding, only the index $i$ of the best matched codeword is transmitted over the channel. In the decoding stage, the decoder merely needs to perform a simple look-up operation from the same codebook for each index $i$ to reconstruct the input vector $\boldsymbol{x}$.

2.2. **The principle of fast codeword search.** In the VQ system, the codeword search problem is vital since it takes most of the encoding time. If we adopt the squared Euclidean distance $d(\boldsymbol{x}, \boldsymbol{c}_i) = \sum_{l=1}^{n} (x_l - c_{il})^2$ to describe the distortion between $\boldsymbol{x}$ and $\boldsymbol{c}_i$, then the full search (FS) of the nearest codeword for each input vector requires $nM$ multiplications, $(2n-1)M$ additions and $M-1$ comparisons. For the VQ system with large codebook size and high dimension, the computation complexity is very high during the codeword search.

To release the computation burden, a lot of fast codeword search algorithms have been presented to speed up the searching process while maintaining the reconstructed performance the same as that of the full search scheme. Among these fast algorithms, one of famous algorithms is called mean-distance-ordered partial codebook search (MPS) [16]. This algorithm is based on the following theorem.

**Theorem 2.1.** *During the nearest codeword search in the codebook $C$ for the $n$-dimensional input vector $\boldsymbol{x}$, assume that the current closest codeword is $\boldsymbol{c}_p$, i.e., $d_{\min} = d(\boldsymbol{x}, \boldsymbol{c}_p)$. For any other codeword $\boldsymbol{c}_i$, if its average value over all components $m_i$ satisfies*

$$|m_i - m_x| \ge \sqrt{\frac{d_{\min}}{n}} \tag{2}$$

*then we have $d(\boldsymbol{x}, \boldsymbol{c}_i) \ge d_{\min}$. Here, $\boldsymbol{c}_i$ and $\boldsymbol{c}_p$ are $n$-dimensional codewords in $C$, $m_i$ is the average value over all components of $\boldsymbol{c}_i$, and $m_x$ is the average value over all components of $\boldsymbol{x}$, i.e.,*

$$\begin{cases} m_i = \frac{1}{n} \sum_{l=1}^{n} c_{il} \\ m_x = \frac{1}{n} \sum_{l=1}^{n} x_l \end{cases} \tag{3}$$

*In other words, if*

$$m_i \le m_x - \sqrt{\frac{d_{\min}}{n}} \quad or \quad m_i \ge m_x + \sqrt{\frac{d_{\min}}{n}} \tag{4}$$

*then the distortion of vector $\boldsymbol{c}_i$ is definitely larger than the current minimal distortion of $\boldsymbol{c}_p$.*

This theorem can be easily proved [16], which means that if we have sorted all the codewords in the ascending order according to their mean values, we can search out the nearest vector by updating $d_{\min}$ iteratively. In the MPS algorithm, the mean value of each codeword is calculated first and then these values are sorted in the ascending order offline. In the encoding stage, the mean value of the input vector is computed, and the codeword that has the absolute difference in the mean value from the input vector is selected as the tentative matching codeword. The squared Euclidean distortion $d_{\min}$ between the input vector and this tentative matching codeword is calculated. For any codeword $\boldsymbol{c}_i$, if Equation (4) is satisfied, then $\boldsymbol{c}_i$ can be eliminated.

2.3. **Gaussian normalization.** In this paper, in order to let each component in the feature vector have the same weight in calculating the distance between feature vectors and thus reduce the effect of one feature component on another feature component, we adopt the Gaussian normalization technique [17] to normalize each component into the same range $[-1, 1]$. Assume there are $M$ images $\{I_0, I_1, \ldots, I_{M-1}\}$ in the database, and we extract an $n$-dimensional feature vector $\boldsymbol{f}_i = (f_{i1}, f_{i2}, \ldots, f_{in})$ from each image $I_i$, where $i = 0, 1, \ldots, M-1$ and $f_{ij}$ means the $j$-th component of the extracted feature vector $\boldsymbol{f}_i$. The normalization equation can be illustrated as follows:

$$c_{ij} = \frac{f_{ij} - a_j}{3\sigma_j} = \begin{cases} \frac{f_{ij}-a_j}{3\sigma_j}, & \left|\frac{f_{ij}-a_j}{3\sigma_j}\right| \le 1 \\ -1, & \frac{f_{ij}-a_j}{3\sigma_j} < -1 \\ 1, & \frac{f_{ij}-a_j}{3\sigma_j} > 1 \end{cases} \quad (i = 0, 1, \ldots, M-1, j = 1, 2, \ldots, n) \tag{5}$$

where $c_{ij}$ is the normalized feature vector component, and

$$\begin{cases} a_j = \frac{1}{M} \sum\limits_{i=0}^{M-1} f_{ij} \\ \sigma_j = \sqrt{\frac{\sum\limits_{i=0}^{M-1} (f_{ij}-a_j)^2}{M-1}} \end{cases} \tag{6}$$

After normalization, the distance between the image $I_i$ and $I_j$ is then measured by the Euclidean distance between their normalized feature vectors $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, i.e.,

$$d(\boldsymbol{c}_i, \boldsymbol{c}_j) = \sum_{l=1}^{n} (c_{il} - c_{jl})^2 \tag{7}$$

3. **Proposed Fast Image Retrieval Scheme.** In this section, the proposed fast retrieval algorithm is given in detail first, and then a simple but efficient method for relevance feedback is explained.

3.1. **Main idea.** To describe our scheme more clearly, we need to give some assumptions. First, we assume that the feature database is composed of $M$ $n$-dimensional feature vectors that are extracted offline from $M$ images in the image database as shown in Figure 3. Second, considering that the user usually only wants to view the first $k$ most similar images ($1 \le k \le M$), and the system interface often can only display $R$ images per page, thus the system needs only to give $P = [(k-1)/R] + 1$ pages of retrieval results, where $[\,]$ denotes the operation to get integer. Third, the squared Euclidean distance measure is used to measure the dissimilarity between two $n$-dimensional vectors as shown in Equation (7).

In the traditional query-by-example CBIR system [18], the similarity comparison is performed between the input feature vector and each feature vector in the image feature database based on a full search method. Different from that, in this paper, we present

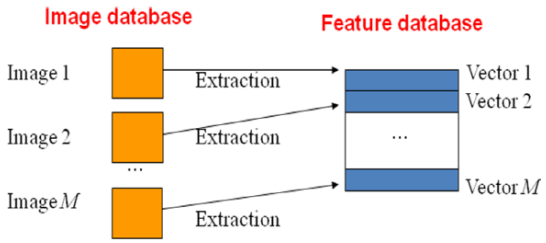


FIGURE 3. Offline feature database construction

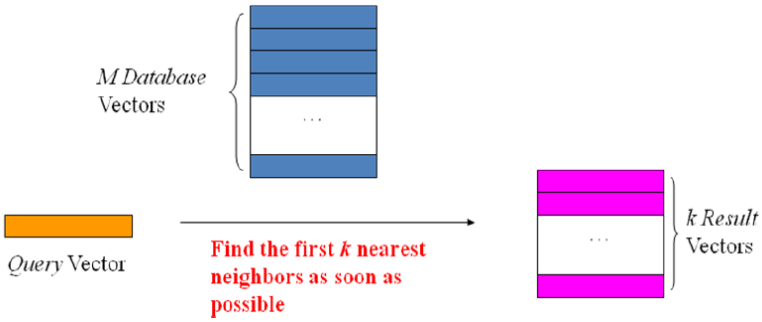


FIGURE 4. The basic search problem in CBIR

a fast retrieval algorithm that can find out the first $k$ most similar images (the $k$ images are displayed in the descending order according to their similarities), and then find out the next $k$ most similar images if necessary. This is a very useful technique since for most users the first several similar images are often just what they need. This is the basic search problem in CBIR as shown in Figure 4.

As we have known, in the MPS algorithm [16], the aim is to find the best-match codeword for the input vector from a codebook. If we try to extend this idea from finding the best-match one to finding the $k$ best-match ones, we can obtain our fast image retrieval scheme named extended fast VQ codeword search (EFVQCS) algorithm. The main idea can be in brief introduced as follows:

If we have obtained $k$ vectors and the vector with the largest distortion from $\boldsymbol{x}$ is $\boldsymbol{c}_p$, we can ensure that there are at least $k$ vectors that have less distortion than those with mean values out of the interval $\left[m_x - \sqrt{\frac{d(x,c_p)}{n}}, m_x + \sqrt{\frac{d(x,c_p)}{n}}\right]$. Thus, the EFVQCS scheme searches candidates near the initial best match vector $\boldsymbol{c}_p$ in an up-and-down manner to find the final $k$ nearest vectors. The detailed EFVQCS algorithm is illustrated in next subsection.

3.2. **Extended fast VQ codeword search algorithm for image retrieval.** Assume we have extracted $M$ feature vectors $\{\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{M-1}\}$ from all the $M$ images in the database and have normalized them into $M$ normalized feature vectors $\{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{M-1}\}$ by Equation (5), and we also have sorted all these normalized feature vectors in the ascending order of their mean values $\{m_0, m_1, \ldots, m_{M-1}\}$. That is, we have obtained the following feature vector set:

$$C = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{M-1}\} \quad m_0 \le m_1 \le \ldots \le m_{M-1} \tag{8}$$

Thus, the proposed fast retrieval algorithm can be illustrated in Figure 5, which can be described as follows:

Input: The query image $X$ and the normalized sorted feature database $C = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{M-1}\}$.

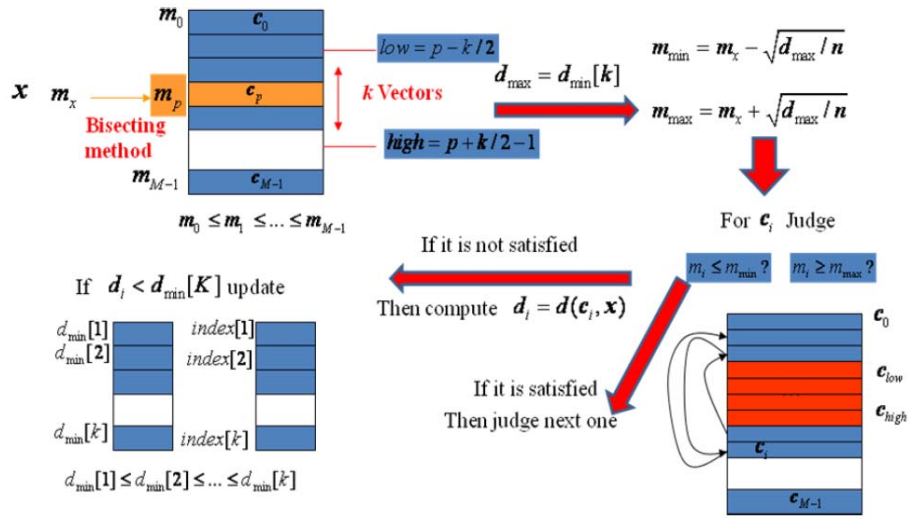


FIGURE 5. The diagrammatic sketch of the EFVQCS algorithm

Output: The first $k$ nearest feature vectors of the query image.

Step 1: Extract the feature vector from $X$ and normalize it using Equation (5), obtaining the query vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. Calculate the mean value of $\boldsymbol{x}$, obtaining $m_x$.

Step 2: Obtain the initial index $p$ of the feature vector with the minimum mean distance from the query vector's mean value by the bisecting method. That is,

$$p = \arg\min_{0 \le j \le M-1} |m_j - m_x| \tag{9}$$

The bisecting method can be described as following substeps:

Step 2.0: Set $i = 0$ and $j = M - 1$, and initialize $p = (i + j)/2$.

Step 2.1: If $i = j$ or $i = j - 1$, go to Step 2.3.

Step 2.2: If $m_x < m_p$, then $j = p$, $p = (i + j)/2$, else $i = p$, $p = (i + j)/2$; go to Step 2.1.

Step 2.3: If $d(\boldsymbol{x}, \boldsymbol{c}_i) < d(\boldsymbol{x}, \boldsymbol{c}_j)$, then $p = i$, else $p = j$; Set $flag1 =$ false, $flag2 =$ false; go to Step 3.

Step 3: Set $low = p - k/2$, $high = p + k/2 - 1$. If $low < 0$, then $low = 0$, $high = k - 1$ and $flag1 =$ true. If $high > M - 1$, then $high = M - 1$, $low = M - k$ and $flag2 =$ true. Sort the data $d(\boldsymbol{x}, \boldsymbol{c}_{low}), d(\boldsymbol{x}, \boldsymbol{c}_{low+1}), \ldots, d(\boldsymbol{x}, \boldsymbol{c}_{high})$ in the ascending order based on the bubble sorting method, and save these distances in the array $d_{\min}[l]$ and their indices in the array $index[l]$, where $l = 1, 2, \cdots, k$.

Step 4: Search the $k$ best matching feature vectors for the input vector $\boldsymbol{x}$. Obviously, we do not need to search the feature vectors with the indices from $low$ to $high$. This step can be described by following substeps.

Step 4.0: Set $j = k/2$. If $low = 0$, then $j = high - p + 1$. If $high = M - 1$, then $j = p - low + 1$. Set $m_{\min} = m_x - \sqrt{\frac{d(\boldsymbol{x}, \boldsymbol{c}_{index[k]})}{n}}$, $m_{\max} = m_x + \sqrt{\frac{d(\boldsymbol{x}, \boldsymbol{c}_{index[k]})}{n}}$.

Step 4.1: If $flag1 =$ true and $flag2 =$ true, then the algorithm can be terminated with the final $k$ indices of most similar vectors, $index[l]$, $l = 1, 2, \ldots, k$.

Step 4.2: If $flag1 =$ true, go to Step 4.3. If $p - j < 0$ or $m_{p-j} < m_{\min}$, then $flag1 =$ true, go to Step 4.3. Otherwise, we compute $d(\boldsymbol{x}, \boldsymbol{c}_{p-j})$, if $d(\boldsymbol{x}, \boldsymbol{c}_{p-j}) < d(\boldsymbol{x}, \boldsymbol{c}_{index[k]})$, then insert $d(\boldsymbol{x}, \boldsymbol{c}_{p-j})$ into the array $d_{\min}[l]$ and the index $p - j$ into the array $index[l]$ in the ascending order of distances, and also update the lower and upper limits of means $m_{\min}$ and $m_{\max}$.

Step 4.3: If $flag2 =$ true, go to Step 4.4. If $p + j \geq M$ or $m_{p+j} > m_{\max}$, then $flag2 =$ true, go to Step 4.4. Otherwise, we compute $d(\boldsymbol{x}, \boldsymbol{c}_{p+j})$, if $d(\boldsymbol{x}, \boldsymbol{c}_{p+j}) < d(\boldsymbol{x}, \boldsymbol{c}_{index[k]})$, then insert $d(\boldsymbol{x}, \boldsymbol{c}_{p+j})$ into the array $d_{\min}[l]$ and the index $p + j$ into the array $index[l]$ in the ascending order of distances, and also update the lower and upper limits of means $m_{\min}$ and $m_{\max}$.

Step 4.4: Set $j = j + 1$, go to Step 4.1.

3.3. **Vector composition based relevance feedback.** Relevance feedback is the step by which a system obtains information from its users about the relevance of features, images, image regions, or partial retrieval results obtained so far. The effect of relevance feedback is to "move" the query in the direction of relevant images away from the non-relevant ones. In CBIR systems with relevance feedback, there are two very important issues. The first one is related to the user interface, i.e., how do we want the users to interact with the system and express their opinion about the image used as an example. While some systems will require minimal action by the user such as telling the system if the results so far are good, bad or neither, others will ask the user to specify numerical values that convey a measure of goodness of those results. The second problem is how to take into account the user's relevance feedback information and translate it into an adjustment on the query, and how to decide the importance of each feature and the probability of each image being the target image.

Up to now, many feedback algorithms have been presented [19,20]. In this paper, we adopt a simple but efficient algorithm for relevance feedback. This algorithm is based on the combination of relevant feature vectors. Assuming we consider $N$ relevant images during each feedback, our scheme can be described as the following equation:

$$\boldsymbol{x}' = \lambda_0 \boldsymbol{x} + \lambda_1 \boldsymbol{c}_{k_1} + \cdots + \lambda_N \boldsymbol{c}_{k_N} \tag{10}$$

where $\boldsymbol{x}'$ is the recomposed query feature vector, $\lambda_0$ is the weighting factor of the current query feature vector $\boldsymbol{x}$, and $\lambda_i$ is the weighting factor of the feature vector $c_{k_i}$ of the relevant image $I_{k_i}$, where $k_i$ is the index of the relevant image, $i = 1, 2, \ldots, N$. The weighting factors should satisfy:

$$\sum_{i=0}^{N} \lambda_i = 1 \tag{11}$$

The above feedback scheme is very simple with low complexity, and if we select flexible $\lambda_i$, this scheme will be very effective.

4. **Experimental Results.** To demonstrate the efficiency of the proposed scheme, we adopt two databases in experiments carried out on a Pentium IV computer with the 2.80GHz CPU. One includes 1000 images [8], which are classified into ten classes, each class including 100 images. The other includes 20000 images of various sizes (without classification). Here we assume the interface can display 16 images per page, i.e., $R = 16$. In our scheme, we adopt multiple features, including the color histogram (the dimension is 250), texture features (the dimension is 3) and shape features (the dimension is 3). Thus the dimension of each feature vector is 256. We perform several experiments to test the effectiveness of the proposed scheme, including the effectiveness of multiple features compared with a single feature, the efficiency of the proposed fast retrieval algorithm compared with the traditional full search method, and the effectiveness of the relevance feedback step.

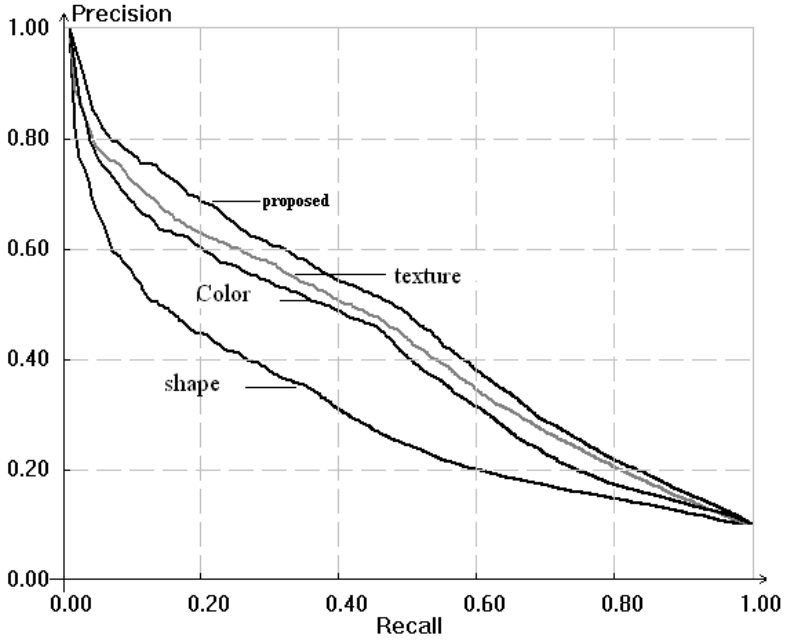


FIGURE 6. P-R curves for various kinds of features

4.1. **Effectiveness of multiple features.** Here, we use the first database with 1000 images to compare the performance of precision and recall among our scheme, the color-histogram based scheme, the texture features based scheme and the shape based scheme. We randomly select 5 images from each class, and thus in total 50 images, as the test query images. For each test query image, we perform the retrieval process based on each kind of features. For each number of returned images (from 1 to 1000), we average the recall and precision value over 50 test query images. Here, the precision and recall are defined as follows:

$$\begin{aligned} \text{precision} &= \frac{\text{No. relevant images}}{\text{No. images returned}} \\ \text{recall} &= \frac{\text{No. relevant images}}{100} \end{aligned} \tag{12}$$

The average P-R curves for various kinds of features are shown in Figure 6. The curve which lies on the upmost is our proposed scheme, which means that our scheme has the best performance. The reason is that our scheme simultaneously consider multiple features.

4.2. **Efficiency of the proposed fast retrieval scheme.** Using the second database with 20000 images, we do the experiment to compare the time performance between the full search (FS) method and the proposed EFVQCS method for different numbers of pages, i.e., $P$ varies from 1 to 10. That is, the ten cases are $k = 16, 32, \ldots, 160$. We randomly select 40 images from the second database to be the test query images. After averaging the required time for each case and each method over 40 test query images, we can obtain the performance chart as shown in Figure 7. From these experimental results, we can see that, when $k$ varies from 16 to 160, $n = 256$ and $M = 20000$, the proposed EFVQCS method only needs about 1% of the required full search time. Here the EFVQCS only needs about 50ms while the FS needs about 4300ms. From Figure 7 we

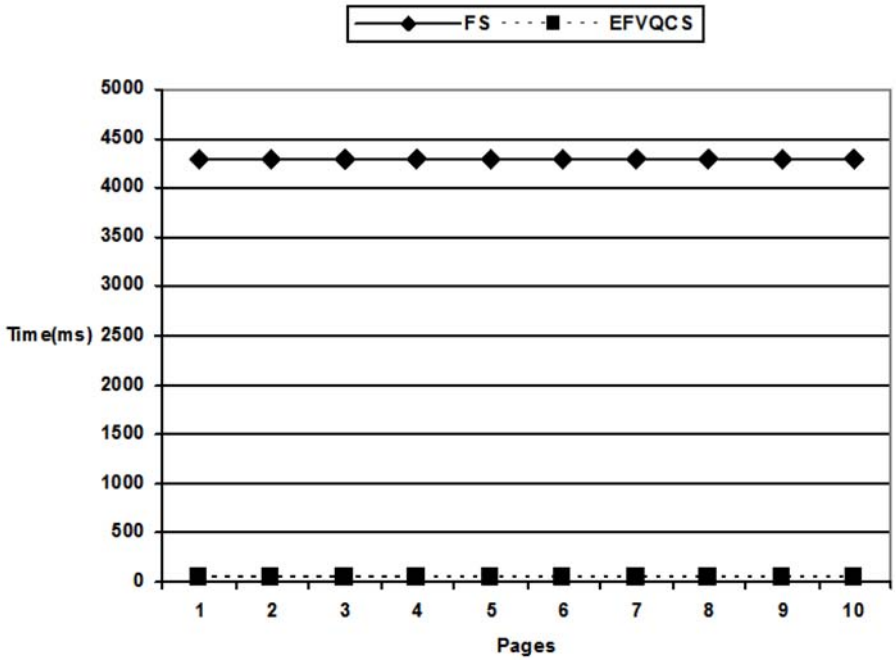


FIGURE 7. Comparison of the time performance between FS and EFVQCS algorithms


FIGURE 8. Retrieval results before feedback


FIGURE 9. Retrieval results after feedback

can also see that the required time for the proposed EFVQCS algorithm nearly linearly

increases with the number of returned pages, which means that we need nearly the same time for each iteration.

4.3. **Effectiveness of relevance feedback.** Here, we use the first database with 1000 images to demonstrate the effectiveness of the proposed relevance feedback method. We show an example in Figure 8 and Figure 9, where the first image in each figure is the query image from the class "people". The experimental results prove that the proposed feedback algorithm is efficient and it improves the precision-recall performance greatly.

5. **Conclusions.** This paper presents a fast retrieval algorithm based on the extended fast VQ codeword search algorithm with a simple effective relevance feedback method. In order to improve the retrieval results, the feature vectors are normalized by the Gaussian normalization technique before retrieval. According to the experimental results, we can see the effectiveness of our retrieval method. The proposed fast retrieval algorithm can speed up the retrieval process extraordinarily, although it costs some time to normalize and sort the vectors offline. Experimental results also demonstrate the practicability of the proposed feedback method. Our scheme can be directly used to many websites for image search to speed up the search speed.

**Acknowledgment.** This work is partially supported by the financial support from the National Natural Science Foundation of China under grant No. 61171150 and the Zhejiang Provincial Natural Science Foundation of China under grant R1110006. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

## REFERENCES

[1] C.-C. Yu, C.-Y. Wen, C.-P. Lu and Y.-F. Chen, The drug tablet image retrieval system based on content-based image retrieval, *International Journal of Innovative Computing, Information and Control*, vol.8, no.7(A), pp.4497-4508, 2012.

[2] L. Cinque, G. Ciocca, S. Levialdi, A. Pellicano and A. R. Schettini, Color-based image retrieval using spatial-chromatic histograms, *Image and Vision Computing*, vol.19, no.13, pp.979-986, 2001.

[3] J. P. K. Kuan, D. W. Joyce and P. H. Lewis, Texture content based retrieval using text descriptions, *SPIE Proc. of Storage and Retrieval for Image and Video Databases VII*, vol.3656, pp.75-85, 1999.

[4] B. S. Manjunath, J. R. Ohm and V. V. Vasudevan, Color and texture description, *IEEE Transactions on Circuits Systems for Video Technology*, vol.11, no.6, pp.703-715, 2001.

[5] K. Lee and W. N. Street, Incremental feature weight learning and its application to a shape based query system, *Pattern Recognition Letters*, vol.23, no.7, pp.865-874, 2002.

[6] S. Panchanathan and C. Huang, Indexing and retrieval of color images using vector quantization, *SPIE Proc. of Applications of Digital Image Processing XXII*, vol.3808, pp.558-568, 1999.

[7] T. Uchiyama, N. Takekawa, H. Kaneko, M. Yamaguchi and N. Ohyama, Multispectral image retrieval using vector quantization, *Proc. of the IEEE International Conference on Image Processing*, vol.1, pp.30-33, 2001.

[8] Z. M. Lu, S. Z. Li and H. Burkhardt, A content-based image retrieval scheme in JPEG compressed domain, *International Journal of Innovative Computing, Information and Control*, vol.2, no.4, pp.831-839, 2006.

[9] Y. Qiao, Z. Lu, C. Zhao and S. Sun, Feature based on modulus maxima of wavelet frame representation for texture retrieval, *International Journal of Innovative Computing, Information and Control*, vol.3, no.6(B), pp.1657-1666, 2007.

[10] A. D. Zhang, B. Cheng, R. S. Acharya and R. P. Menon, Comparison of wavelet transforms and fractal coding in texture-based image retrieval, *Proc. of Visual Data Exploration and Analysis III*, vol.2656, pp.116-125, 1996.

[11] F. X. Yu, H. Luo and Z. M. Lu, Colour image retrieval using pattern co-occurrence matrices based on BTC and VQ, *Electronics Letters*, vol.47, no.2, pp.100-101, 2011.

[12] Y. Linde, A. Buzo and R. M. Gray, An algorithm for vector quantizer design, *IEEE Transactions on Communications*, vol.28, no.1, pp.84-95, 1980.

[13] F. Idris and S. Panchanathan, Storage and retrieval of compressed images, *IEEE Transactions on Consumer Electronics*, vol.41, no.3, pp.937-941, 1995.

[14] Z. M. Lu and H. Burkhardt, Colour image retrieval based on DCT-domain vector quantisation index histograms, *Electronics Letters*, vol.41, no.17, pp.956-957, 2005.

[15] W.-M. Zheng, Z.-M. Lu and H. Burkhardt, Color image retrieval schemes using index histograms based on various spatial-domain vector quantizers, *International Journal of Innovative Computing, Information and Control*, vol.2, no.6, pp.1317-1326, 2006.

[16] S. W. Ra and J. K. Kim, Fast mean-distance-ordered partial codebook search algorithm for image vector quantization, *IEEE Transactions on Circuits Systems II*, vol.40, no.9, pp.576-579, 1993.

[17] Q. Iqbal and J. K. Aggarwal, Combining structure, color and texture for image retrieval: A performance evaluation, *Proc. of the International Conference on Pattern Recognition*, vol.2, pp.438-443, 2002.

[18] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin, The QBIC project: Querying images by content using color, texture and shape, *Proc. of SPIE Storage and Retrieval for Image and Video Databases*, pp.173-187, 1993.

[19] D. Wu, D. He and H. Wang, A relevance feedback based query translation enhancement technique in cross language information retrieval, *Journal of the China Society for Scientific and Technical Information*, vol.31, no.4, pp.398-406, 2012.

[20] J. H. Su, W. J. Huang, P. S. Yu and V. S. Tseng, Efficient relevance feedback for content-based image retrieval by mining user navigation patterns, *IEEE Transactions on Knowledge and Data Engineering*, vol.23, no.3, pp.360-372, 2011.